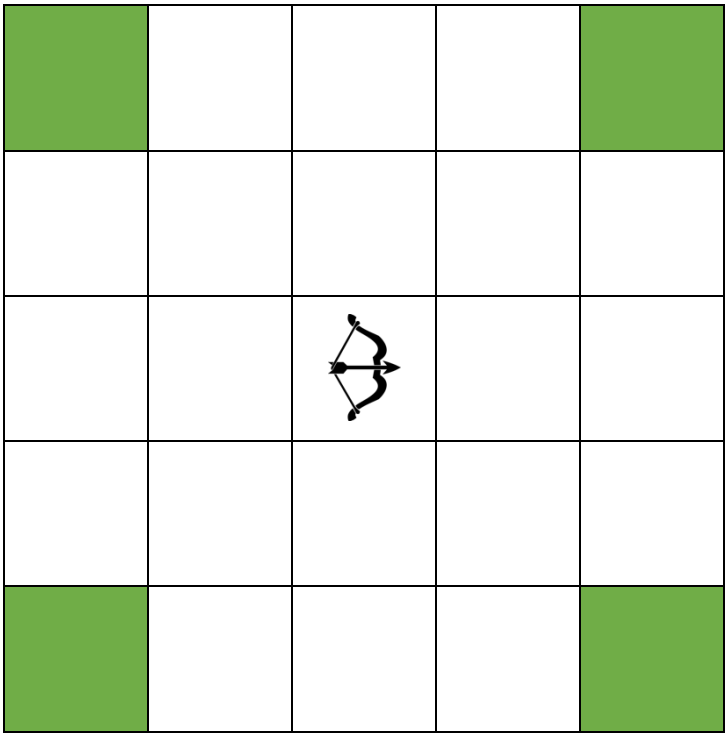
5. Harish

Program Name: Harish.java **Input File: harish.dat**

Harish is an avid Uber Intense Life board game player. Uber Intense Life, is a game similar to chess, in which players must use pieces such as the archer, squire, warrior, and blacksmith to surround the opponent's capitol and force a surrender all while protecting their own capital from enemy invasion. Harish's favorite Uber Intense Life piece is the archer, a piece known for its long-range attack. A legal attack for an archer to make is by shooting an arrow 2 units either up or down, and 2 units either left or right. An archer cannot shoot an arrow that requires the arrow to exit the board. The below figure shows an example of the four possible attacks an archer can make. The shaded squares show valid squares for the centralized archer to attack.



Harish has the idea to try and place **nine** archers on a 5x5 board such that no two archers could attack each other, i.e., no archer could fire an arrow into an already occupied square. The problem is, Harish has no way to verify if a given configuration of archers meets the above criteria. Can you help him write a program to verify that nine distinct archers are placed on a 5x5 board and that no two archers can attack each other?

Input: Input begins with an integer N (1 <= N <= 20), the number of different test cases. Each test case will consist of a 5x5 grid of characters. An "a" represents an archer's location and a "." represents an empty, unoccupied square. Each test case will be followed by a line containing ten dashes (-).

Output: For each test case, your program is to output `valid` if and only if **nine** archers are placed on the board such that no two archers could attack each other, otherwise your program should output `invalid`.

Sample input and output are on the next page to avoid a page break in the middle of input.

Harish continued:

Sample Input:

```
6
...a.
...a.
a.a..
.a.a.
a.a.a
----------
.a...
...a.
a...a
.a.a.
a...a
----------
.a.a.
....a
a...a
.a.a.
a...a
----------
.a.a.
a.a.a
.....
.....
aaa.a
----------
a.a.a
.a.a.
.....
.a...
a.a.a
----------
a..a.
.a..a
a..a.
a....
a...a
----------
```

Sample Output:

```
invalid
invalid
valid
valid
invalid
valid
```